SOLUTIONS - CS110 Review Questions for Final Exam

Review Questions...

Q1

A. statement
B. strings
C. semi-colon
D. src
E. logical
F. local variable

Q2:

Variables store values in the memory during program execution. We need them the most if we want to use such values many times within a program.

Initially, variables are used to store some input information [Gathering Phase]. This input is either given interactively by the user (maybe through a form) or is already known (we can have images for a gallery, items to sell, etc.).

Then, variables participate in different expressions (arithmetic, logical, relational) or function invocations in which the information is processed or decisions for the flow of the program are made.

Finally variables are used to convey the results of such processing [Output Phase] by giving feedback to the user or changing the appearance of the page.

Q3:

“undefined” is the default value that JavaScript assigns to variables which were not assigned a value by the program. However, when you see this value in the console, it is usually because you are executing a statement that performs an action without evaluating to a value.

Q4:

No.
A relational operator is used to **compare** two expressions that evaluate to some primitive value (numerical, string, boolean) such as:

```
// expressions on both sides of operator evaluate to numbers
3 + 5 > 2 * 4

// expressions on both sides of operator evaluate to strings
"sa" + "m" != "SAM"
```

```
// expressions on both sides of operator evaluate to booleans
isRaining == false
```

A logical operator is used to **combine** two relational expressions to create more complex expressions, known as logical expressions. In this case on both sides of the logical operator there must be expressions that evaluate to a boolean value.

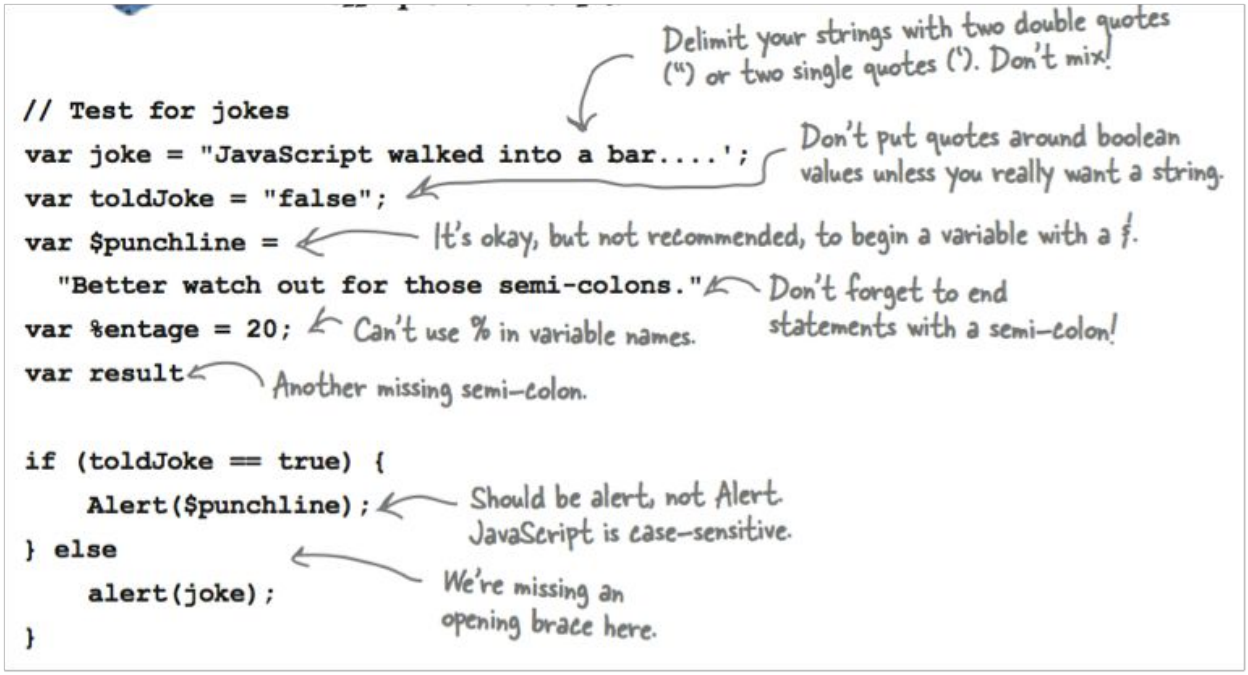
To see what happens when you try to use a logical operator in the same kind of expressions as a relational operator execute the following code in the console, **by entering it line by line** to see the result every time.

```
var a = 10;
var b = 2;
a > b; // Evaluates to a boolean value.
a + 3 && b * 4; // WRONG use. Evaluates to a number.
a > 5 && b < 4; //CORRECT use. Evaluates to a boolean value.
```

Q5:

A. `humid` is `true`
B. `isValid` is `true`
C. `sendFile` is `true`

Q6:



Q7:
Here is an example statement that declares an array of string values:

```
var studentNames = ["Harry Potter", "Hermione Granger", "Ron Weasley"];
```

`var` is a reserved JavaScript word (a keyword) that always starts a variable declaration statement.

`studentNames` is the name for the variable. We can name a variable whatever we like, but usually is preferable to assign a name that conveys the meaning of the values to be stored in

= is the assignment operator. It's a kind of a binary operator because it has operand in both sides.

`[ ]` is the notation for an array literal; it tells Javascript that we are storing a value of type array.

`"Harry Potter", "Hermione Granger", "Ron Weasley"` are the elements of the array; they must be separated by commas.

Q8:
The location in the program where we define a variable influences its visibility by other parts of the program.

If a variable is defined outside a function, it's *global*; if it's declared inside a function, it's *local*.

```
// the global variable slideshow_index is
// declared outside of any function definition
var slideshow_index = 0;

function playSlideshow() {
      // the local variable done is declared
      // inside the function definition
      var done = false;
}
```

Q9:
Local variables are visible (or exist) only within the body of the function where they were defined.

Q10:

`setInterval` is a built in JavaScript method that accepts two parameters: 1) A function and 2) A time in milliseconds.

Once invoked, setInterval will execute the given function (A) every x milliseconds specified (B).

In this example, **`stillThere`** is the function and it will be invoked every 2000 milliseconds (i.e. every 2 seconds).

New Topics

Q11:
The recording quality will improve when you use a higher resolution, but the file size will also increase. For example, using a resolution of 8 bits means an 8-bit value is used to store each sound sample. But that only gives 2^8=256 distinct levels that a sound sample can represent. The bit-resolution of CD-quality music is 16: so, 16-bit numbers are used to store the value for each sample, giving us 2^{16}=65,536 distinct levels from lowest to highest. So, changing from 8 to 16 bit resolution should definitely produce a better quality recording. However, the size of the file for the given recording length will also double by changing from 8 to 16 bit resolution, because each sample uses twice as many bits.

Q12:
Bit rate = bit resolution * sampling rate
File size = bit rate * length of recording

CD-quality music is sampled at 44KHz, because humans can hear up to 22KHz (and according to the Nyquist Theorem, you should sample at twice the highest frequency you wish to capture).

So, If you use an 8-bit/1-byte resolution, 10MBytes = 1 byte/sample * 44KHz * length, so length = 227.2 seconds (3.78 minutes, but this will probably be unacceptably low quality).

Using 16-bit resolution/2-byte resolution (CD-quality), 10MB = 2 bytes/sample * 44KHz * length, so length = 113.6 seconds. (1.9 minutes).

Q13:
SHOW ME THE MONEY

The easiest way to solve this is to try out the possible two-letter combinations for the second word until you find a rotation that matches both letters. Then, it is easy to apply that rotation to all the other letters of the phrase.

When the length of the individual words is not known, Caesar ciphers can be solved through frequency analysis (certain letters in the English language occur much more frequently than others).

Q14:
Apply the keyword to the cipher, then use the rotations table from lecture to decipher the message.

CWGZSK OG VUABTU
GOTGOT GO TGOTGO

WINTER IS COMING

Q15:
They are both examples of ***private key cryptography***, because in each case, the two parties who are communicating must both have the secret “key” (rotation for Caesar cipher, keyword for Vigenere cipher) to either encrypt or decrypt messages.

Q16:
If Bob wants to receive secure messages, he need to have a key pair. Alice needs to have Bob's public key in order to send him a message that only he can read.

Q17:
He can now decrypt messages sent to her. He is also able to forge her digital signature, thereby sending messages that purport to be from her.

Q18:
It’s digital because it’s not proportional to any physical property of the input, and it’s easy to clean up (a dot that’s a bit too long, or a dash that’s a bit too short, should still be easy to distinguish.)

Q19:
Your browser will tell you who signed the certificate and whether you would like it to trust the certificate once, never, or always. You can also choose to be warned before transmitting data to the site. It’s up to you whether to trust the certificate and how much.

Note: one possible misinterpretation of the question might be “there is no signed certificate” – that’s different from having a certificate that isn’t signed by a recognized authority.

Q20:
You need to watch out for two red flags: If the site uses http rather than https, then you shouldn’t enter your credit card number because it will be sent unencrypted. If the message about digital certificate signed by unrecognized authority pops up, that’s an indication that the website might

be not legitimate. The site might store your credit card number on the computer as a cookie, which would be bad if it were not your own computer.
Here's one scenario where the URL might start with http but contain a *frame* whose URL was https. Finally, of course, all the digital security in the world would not prevent the actual vendors from being thieves.

JavaScript

Note: The following are examples of possible solutions; there may be other correct solutions as well.

Q21:
Code for weather.html
(note doctype and charset are omitted for brevity)

```
<html>
	<head>
	<title>Weather page</title>
	<script src="weather.js"></script>
</head>

<body>
	<p>
		<img id="weather" src="" alt="seasonal image">
	</p>
</body>
</html>
```

Code for `weather.js` (this solution does not use jQuery):

```
document.onload = selectImage;

function selectImage() {
	var today = new Date();
	var month = today.getMonth();
	var url   = "summer.gif";

	// if month is December or before April
	if (month == 11 || month < 3) {
		url = "winter.gif";
	}

	document.querySelector("#weather").src = url;
}
```

Q22:
(This solution uses jQuery):

Code for splash.html

```
<html>
	<head>
		<title>Splash page</title>
		<script
		src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js”></script>
		<script src=”random.js”></script>
	</head>

<body>
	<header>
		<img src=”” alt=”random image”>
	</header>
	</body>
</html>
```

Code for random.js:

```
var images = [“image1.gif”,”image2.gif”,”image3.gif”];

var num = Math.floor(Math.random() * 3);
$(“header img”).attr(“src”,images[num]);
```

Q23:
A.

```
var person = {
    toddler: {
        start: 0,
        end: 2
    },
    child: {
        start: 3,
        end: 11
    },
    adolescent: {
        start: 12,
        end: 17
```

```
    },
    adult: {
        start: 18,
        end: 120
    }
}
```

B. omitted

Q24:
Code for `age.html`:

```
<html>
	<head>
		<title>Age page</title>
		<script
	src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js”</s
	cript>
		<script src=”age.js”></script>
	</head>

<body>
	<p>
		<label>Enter age <input id=”age” type=”text”></input></label>
		<button id=”find_category” >Calculate</button>
	</p>

	<div id=”show_category”></div>
</body>
</html>
```

Code to add to `age.js`:

```
$(“#find_category”).click(function () {
	var age = parseInt($(“#age”).val ());
	var category = findCategory(age);
	$(“#show_category”).text(“You are a “ + category + “.”);
});
```

Q25:
The following function is fine:

```
function isLarger(val1, val2) {
    if( val1 > val2 ) {
```

```
        return true;
    } else {
        return false;
    }
}
```

The following function is equivalent but more concise. Do you see that it is equivalent?

```
function isLarger(val1,val2) {
      return val1 > val2;
}
```

Q26:

Code for `compare.html`:

```
      <html>
            <head>
                  <title>Compare page</title>
                  <script
      src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js
      ”</script>
                  <script src=”compare.js”></script>
            </head>

      <body>
      <p>
            <label>Enter value 1<input” type=”number”></input></label>
            <label>Enter value 2<input” type=”number”></input></label>
            <button>Compare</button>
      </p>

      <ol>
      </ol>

      </body>
      </html>
```

Code to add to `compare.js`:

```
      $("button").click(function() {
            var val1= $("[type='number']")[0].value;
            var val2= $("[type='number']")[1].value;
```

```
            var result = isLarger(val1,val2);

            $(“<li>”).text(val1 + " > " + val2 + " = " + result).
            appendTo(“ol”);
      });
```

Q27:
Code for car.js:

```
function thumbClick() {
    var bigurl = $(this).attr("data-big");
    $("#car").attr("src",bigurl);
}

$(".thumb").click(thumbClick);
```

CSS

Q28:
Create a class disguised, and assign to our links:

```
      a:link.disguised {
            color: black;
            text-decoration: none;
      }
      a:visited.disguised {
            color: black;
            text-decoration: none;
      }
```

This would be used like:

```
      <a class="disguised" href="http://secretlocation.com">link</a>
```

Q29:
As with the last answer, we could use a class, but it’s annoying to have to put in all those class=... attributes. Instead, we can use nested selectors:

```
#navbar a:link {
      color: red;
      text-decoration: none;
}
```

Then, our navbar should have that id:

```
<div id="navbar">
      <a href="home.html">home page</a><br>
      <a href="about.html">about our site</a><br>
      <a href="links.html">links</a><br>
</div>
```

Q30:
The “clear” property says where floating elements are not allowed. If we say clear: left on an element, then no floats can appear to its left, which means that all the floating elements that precede it (in the markup) will be above it on the page: the element will move down as far as necessary so that all the floats will be out of the way

Q31:
Width does not include the width of padding, borders, or margins. Therefore, the inner div takes up 340=300+20+20 pixels horizontally.

To the outside of the red border, the outer div takes up 320=300+10+10 pixels horizontally. So, the inner div doesn’t fit inside the outer one.

Q32:

```
.evenStyle {
      color: white;
      background-color: green;
}

.oddStyle {
      color: green;
      background-color: white;
}
```

Q33:

```
var today = new Date();
var todaysdate = today.getDate();

if (todaysdate%2 == 0) {
      $(“body”).addClass(“evenStyle”);
} else {
```

```
        $(“body”).addClass(“oddStyle”);
}
```